1 Performance Tuning

Contents

This section provides performance tuning tips for:

- Oracle Web Application Server
- The Oracle Database

Please check

```
http://www.olab.com/
```

for updates on performance tuning.

Oracle Web Application Server

This sections provides information about tuning the Web Request Broker and cartridges.

Minimum and Maximum Number of Cartridges

You can maximize the performance of Oracle Web Application Server by properly adjusting the minimum and maximum number of cartridges based on the average and peak connection rate. The ideal minimum is 60%-80% of concurrent connections which can be expected on your server. You must take into account the amount of memory available. Ideally, there should be enough cartridges to allocate one for each concurrent request. But due to memory constraints, this may not always be possible. A properly tuned server balances the number of requests served with performance, based on memory used. Here is an example:

```
Max. number of concurrent connections expected = 25
Recommended maximum = 30
Recommended minimum = 15-20
```

The sum of memory required by the maximum of all cartridges needs to be less than the amount of memory available. In general, it is advisable that when all the cartridges are up and running, the usage should not exceed 75% of available swap space.

Spare Cartridges

A new feature to the 3.0.1 release is the ability to have spare cartridges running. This feature improves performance when there is a high volume of requests for a particular cartridge type. Prior to the 3.0.1 release, if a cartridge type received more requests than it could handle, the dispatcher requested a new cartridge, placing the request on hold until the new cartridge became available. The “Spare Cartridge” feature reduces the effect of this performance problem by having a pre-determined number of each cartridge type available.

The following algorithm determines the number of spare cartridges:

```
spareK = MIN ((maxK - minK)/spareKRatio, maxSpareK)
```

where:

- **`spareK`** = the number of spare cartridges for a particular cartridge type
- **`MIN`** = whichever is lesser of what follows
- **`maxK`** = user defined maximum number of cartridge instances for a particular cartridge type
- **`minK`** = user defined minimum number of cartridge instances for a particular cartridge type
- **`spareKRatio`** = user defined environment variable `OWS_BROKER_SPAREK_RATIO` which adjusts the number of spare cartridges to the volume of requests (default is 10 for all cartridge types)
- **`maxSpareK`** = user defined environment variable `OWS_BROKER_MAXSPAREK` which sets a maximum number of spare cartridge instances (default is 5 for all cartridge types)

For example:

`maxK`	`minK`	`spareKRatio`	`maxSpareK`	**Spare Cartridges**
30	10	10 (default)	5 (default)	2
30	10	4	5 (default)	5
30	10	1	5 (default)	5
50	10	5	10	8
300	0	10	10	10

While setting the environment variables, take the following into account:

- The Spare Cartridge feature can be disabled by setting the `OWS_BROKER_MAXSPAREK` environment variable to `0`.
- Allowing a high number of spare cartridge instances can use up swap space, database connections, and CPU time.
- The number of cartridge instances of a particular cartridge type never exceeds `maxK`, in spite of any spare cartridges.

Setting Timeout for Idle Cartridges

The Dispatcher serves incoming requests by allocating a cartridge to serve that request. If no cartridge is available, the WRB allocates a new cartridge.

Both the Dispatcher and WRB have heartbeat (HB) routines that are set by environment variables which determine their timeout. The Dispatcher sends idle cartridges back to the WRB once the number of seconds set for OWS_ORAWEB_MINMAX is reached. The WRB deactivates any extra cartridges one second plus the number of seconds set for OWS_BROKER_HBFREQ and times out when that time is reached. If the number of idle cartridges exceeds the minimum number set, the WRB terminates the extra cartridges.

To change the timeout of idle cartridges, you can:

- Set the environment variable OWS_ORAWEB_MINMAX to a smaller value to increase the frequency of timeout, or a larger value to decrease the timeout.
- Change the frequency the WRB destroys extra cartridges by changing the environment variable, OWS_BROKER_HBFREQ, to either a higher or

lower number, depending on whether you want to increase or decrease the frequency of deactivation.

Note: You must restart the WRB after changing the environment variables.

Multiple Processes for a Listener

In release 3.0 and earlier of Oracle Web Application Server, if you use the Oracle listener (as opposed to listeners from other vendors such as Netscape or Microsoft), each listener was associated with two processes: one to handle incoming requests, and another to perform DNS resolution.

In release 3.0.1 and later, you can configure each Oracle listener to have multiple processes to handle incoming requests. Each listener has the following processes:

- one process to perform DNS resolution
- a configurable number of processes to handle requests (“listener processes”). The default is 1.
- one master process that monitors the above processes

These are all **oraweb** processes.

Having multiple listener processes makes Oracle Web Application Server more robust and reduces the time that clients are unable to connect to the server. If one of the listener processes goes down for any reason, incoming requests are handled by the other listener processes. In addition, the master process starts up another listener process so that the specified number of listener processes is always maintained, and the **wrbdm** (dispatch monitor) process cleans up the cartridge processes that were associated with the listener that went down.

Note that the **wrbdm** process also cleans up cartridges for non-Oracle listeners (for example, Netscape and Microsoft) when the listeners go down for any reason.

Having more than one listener process also improves performance because there are more processes to handle incoming requests. This is especially true if you are running Oracle Web Application Server on a multi-processor machine. Oracle recommends that you set the number of listener processes to the number of CPUs on your machine.

To change the default of one listener process, use the Oracle Web Application Server Manager.

Configuration file detail: The number of listener processes is set in the `SpyglassProcesses` directive in the `Server` section of the **sv<listenerName>.cfg** file. For example:

```
[Server]
  SpyglassProcesses = 4
```

Cartridge Buffer Size

The cartridge internal buffer size (in bytes) has a default value of 60,000 bytes. The most important criteria for this parameter is that it must be big enough to hold all the headers written through WRBClientWrite. The larger the size, the better the performance.

Logging

Each time a request is serviced, information is logged to the listener log file. If the log file is on the same disk as the document root, there is contention for the disk between request reads and log writes. Oracle strongly recommends that the log file be placed on a separate disk. Also, on a stable system, the logging level should be set to a minimum value (1). Refer to the Oracle Web Application Server documentation for detailed information on logging.

Monitor Errors

It is a good idea to regularly monitor the errors generated in the server. Large number of errors in the server waste resources on the system and may be easily fixed by correcting the configuration. For example:

- HTTP error 404 is usually caused by a faulty link.
- Error 500 might be caused by an image map with holes in the coordinates and a user clicks when the pointer is over an undefined region. To remedy this, you should have a default entry for every image map. This directs the server to a default URL if the user clicks anywhere outside of the defined region.

Spyglass HTTP Server

This section contains tuning information which is specific to the Spyglass HTTP Listener which is bundled with Oracle Web Application Server 3.0.

Redirection

Specifies the maximum number of requests (up to 338) that can be handled concurrently by the listener. If a listener's maximum number of requests is set at 100, then the 101st request is sent back to the browser with a redirection status and the URL of the new destination. The browser interprets this status and sends a request to the new destination. This operation is transparent to the user.

In many situations, redirecting the request to a secondary listener may be better than waiting for service on a loaded primary listener. Since the redirection adds the cost of a network round-trip, the needs of an Intranet are obviously different than for the Internet.

For example, if the maximum of requests for Listener1 was set to 100, then it would redirect concurrent requests beyond 100 to Listener2. The question is: what is more expensive, a request that makes a round trip to the browser and back to the new server or waiting for the primary listener to service this request along with 100 other concurrent requests? Here is an example:

```
Expected peak number of concurrent requests = 150

Possible redirection point = tuned between 75-125
```

Rescan Intervals

The listener maintains a cache of all the directory entries under DOC_ROOT. By default, for every request it verifies the file update status by confirming it in the directory entry. The Rescan Interval parameter specifies how often (in seconds) the DOC_ROOT directory structure is scanned. For sites where the content changes infrequently, this rescan interval should be set to a high value. For example: If the Rescan Interval parameter is set to 3600 (1 hour), the directory will be rescanned every hour. Any changes in files in DOC_ROOT will not be picked up for a maximum of one hour, unless the server is restarted or reloaded. When a request is received, the listener must sort through its directory structure to find the file.

Keeping a large number of files in a single directory will slow down this search each time. Instead, segregate files logically into separate directories.

DNS Resolution

Each time a request is received, the IP address received from the client is translated into a Domain Name Service hostname and logged in the log file. This can be an expensive operation, especially if the DNS server is heavily loaded. The DNS name resolution is controlled by the DNS Resolution parameter in the listener configuration file (svlistener_name.cfg). It can be set to the following values:

- **ALWAYS -** Always translate IP addresses into DNS hostnames.
- **LAZY -** Translates IP addresses into DNS hostnames only when needed by a Spyglass ADI application or security module.
- **LAZY_WITH_CGI -** Translates IP addresses into DNS hostnames only when needed by CGI programs.
- **NEVER -** Never use DNS hostnames (default).

Due to the added cost of this operation, Oracle recommends that you set this parameter to NEVER. If DNS hostnames are required for logging and tracking purposes, the DNS reverse name-lookup off-line. This can be accomplished by running a script at periodic intervals to go through the archived log file and replace the client IP addresses with the hostnames by performing a DNS reverse lookup at the DNS server.

The Oracle Database

Multi-Threaded Server

Ideally, dedicated Oracle server processes should be allocated for all the PLSQL cartridges. But for sites that run into memory and swap space constraints, Oracle recommends that you use the Oracle Multi-Threaded Server. It is an effective way to multiplex your connections to the database, ensuring that each server process is consistently being used to the maximum.

Please refer to the Oracle Database documentation for more details on architecture, installation and configuration of MTS.

Database Connections

Each PLSQL cartridge, upon startup, tries to connect to every database that is specified in the owa.cfg file. This can slow down the startup process. It is recommended that you keep the entries in the owa.cfg to a minimum. For a large number of database connections, you should partition the connections into logical groups. Please refer to the documentation on the PLSQL cartridge for more information on partitioning.